CSC3100 Assignment1 Report

Question 1

Problem statement:
The problem is to find the number of disorder pairs in a queue where people are supposed to stand in ascending order of their heights. A disorder pair is defined as a pair of people (pi, pj) such that i < j and pi is taller than pj in the queue, indicating their order is reversed.
To solve this problem, we need to find the compare the height of different individuals in the queue. It is similar to the sorting algorithms introduced in lectures, so we can add the count-the-disordered-pair step in different sorting program to get the result.

Solution:
My solution is based on merge sort program.
In the “merge” function, the algorithm counts inversions. When merging two sorted sub-queues, if an element from the right sub-queue is smaller than an element from the left sub-queue, it indicates an inversion. Because the sub-queues are already sorted, if an element from the right sub-queue is smaller, it's also smaller than all the remaining elements in the left sub-queue.
When an inversion is found, the count is increased by the number of remaining elements in the left sub-queue (mid - i + 1). This addition accounts for all the elements in the left sub-queue that are greater than the current element from the right sub-queue. Then add the inversions from all merges together in the “mergeSortAndCount” function and call this function in main function to get the total disorder pairs.
Other solutions include based on insertion sort. This compare every pair of people in the queue, and has a time complexity of O(n^2), which is not efficient for large input sizes. My solution used the idea of divide and conquer, and has the time complexity of O(n*logn), more efficient for large input size.

Question 2

Problem statement
The stars in nth field fn is denoted by fn = a*fn-1 + b*fn-2, and the problem need us to find fn mod m since fn can be too large (a, b, m given). Based on the hint3, this problem can be transformed to the power of a certain matrix {[a, b],[1, 0]}, because we can multiply the result with the vector (f1, f0) to get the final result fn.
Meanwhile we want only the fn%m, not fn, we can directly calculate the modulo using the equation given in hint2.

Solution:

We can solve the problem with brute force by multiply all the matrix one by one. That way has the time complexity of O(n).
My solution uses idea of calculating power quickly. For example, if we calculate A^2, we can get A^4 by getting power A^2, instead of multiply A^2 by A for 2 times. That is to say, when calculate A^n, we can calculate the power of A^(n/2), and A^(n/2) is the power of A(n^4), etc.. I first defined matrix multiply with computing the modulo at the same time in the “matrix_mult” function, so i can use it to multiply matrix and getting modulo like multiply numbers. Then in the matrix power function, I used a recursion to compute power of matrix and compute the mod, with the idea mentioned before. Then in the “calculate” function, I multiplied the matrix with (f1, f0) vector and get the mod.
My solution would have a time complexity of O(logn), more efficient than bruteforce.